SATBAYEV UNIVERSITY

Computing & Engineering

Volume 1 (2023), Issue 1, 13-19

https://doi.org/10.51301/ce.2023.i1.03

Using machine learning algorithms for processing medical data

G. Mukazhanova[1], Zh. Alibiyeva[2*], A. Kassenkhan[2], N. Mukazhanov[2]

[1]*Innovative Eurasian University, Pavlodar, Kazakhstan*
[2]*Satbayev University, Almaty, Kazakhstan*

**Corresponding author: zh.alibiyeva@satbayev.university*

Abstract. The paper considers the comparative analyses of machine learning algorithms for dataset: cardio_train.csv from kaggle.com (link: https://www.kaggle.com/sulianova/cardiovascular-disease-dataset). Moreover, using machine learning algorithms there will be discovered the best accuracy algorithms for the cardio_train.csv. Considering procedures have done in Python 3.0 programming language, which represents confusion matrix and classification report, in order to see precision score, recall, f1-score, and support. Furthermore, in this paper you are able to see following classification models: KNN algorithm, Logistic Regression, Decision Tree, Random Forest, Naïve Bayes and SVM. As a result, it will be defined the superior accuracy for processing medical dataset.

Keywords: *medical dataset, machine learning, algorithms, KNN algorithm, Logistic Regression, Decision Tree, Random Forest, SVM, Naïve Bayes.*

1. Introduction

It is clear, that processing medical data plays significant role in our century, especially in period of worldwide pandemic which has changed and affected to world health organization and whole economy of the countries. The mission of this paper is to help and to analyze medical data via modern technology such as machine learning and to process cardiovascular diseases via finding out methods and models in order to atomize data and compare methods, search the best-adapted models and method using Python programming language and Machine learning algorithms. There is a good medicine in Kazakhstan, although it needs several methods in order to make it better. The purpose of the following work is to improve processing medical data, especially cardiovascular diseases, which is the top problem in Kazakhstan.

In this article, will be presented machine learning classification algorithms such as: KNN algorithm, Logistic Regression, SVM, Decision Tree, Random Forest, Naïve Bayes. All algorithms and procedures have done in Jupyter Notebook (Anaconda), Python 3.0 programming language. After considering machine learning algorithms will provided comparative analysis in tables of all procedures.

2. Materials and methods

2.1. Overview of dataset

There are 3 types of input features: objective, examination and subjective.

Objective: factual information;

Examination: results of medical examination;

Subjective: information given by the patient.

Features correspond to 12 columns: age, height, weight, gender, systolic blood pressure, diastolic blood pressure, cholesterol, glucose, smoking, alcohol intake, physical activity and presence or absence of cardiovascular disease.

Below are the column details:

Age | Objective Feature | age | int (days)

Height | Objective Feature | height | int (cm) |

Weight | Objective Feature | weight | float (kg) |

Gender | Objective Feature | gender | categorical code |

Systolic blood pressure | Examination Feature | ap_hi | int |

Diastolic blood pressure | Examination Feature | ap_lo | int |

Cholesterol | Examination Feature | cholesterol | 1: normal, 2: above normal, 3: well above normal |

Glucose | Examination Feature | gluc | 1: normal, 2: above normal, 3: well above normal |

Smoking | Subjective Feature | smoke | binary |

Alcohol intake | Subjective Feature | alco | binary |

Physical activity | Subjective Feature | active | binary |

Presence or absence of cardiovascular disease | Target Variable | cardio | binary | All of the dataset values were collected at the moment of medical examination.

All the following attributes will help to analyze and process the best adapted method and models in order to achieve in the article goal.

Cardiovascular disease (CVD) continue to be the most pressing health problem most countries of the world, including the Kazakhstan. According to the World Health Organization, every year in the world from cardiovascular disease (CVD) dies more than 17 million people, including more than 7 million from coronary heart disease (IHD) [1].

Predicting CVD risk is becoming increasingly more important in clinical decision making since their introduction at the international level in the latest guidelines. At the same time, predicting the risk of coronary artery disease at based on the analysis of traditional risk factors is fraught with a number of problems. In the FI, during observation for 26 years, a significant coincidence of groups of persons without established ischemic heart disease and people who develop coronary artery disease. By level traditional FRs, coincidence

was noted the level of total cholesterol (3.9-7.8 mmol / l) between the groups [2].

There was a significant overlap of groups of patients IHD and healthy men according to the level of traditional RF (TC, LDL cholesterol, smoking, AH, BMI) and significant difference in HDL cholesterol, TG and ratio LDL cholesterol / HDL cholesterol. The prospective NPHS2 study compared the predictive ability of algorithms for cardiovascular risk assessment by Framingham and Procam. Both of these algorithms had a false negative result > 85%.

The low accuracy of predicting cardiovascular events has a number of reasons. First, the assessment the total risk must be adapted depending on national and regional features. Secondly, considering the design of the research scales included in the development, in them often not considered significant for the offensive cardiovascular event's clinical conditions (type I and II diabetes mellitus, chronic kidney disease or very high levels of certain risk factors). Third, the data that were used to compile scales, were received 30-50 years ago and may not correspond to modern realities. Fourthly, mathematical methods for calculating risks also have errors and limitations of applicability. This way we can speak with confidence about the problem of insufficient accuracy of results calculating cardiovascular risk based on generally accepted scales.

Machine learning provides good opportunities to solve this problem and significantly improve accuracy in predicting cardiovascular diseases and their complications in comparison with the use of existing methods, due to the nonlinear relationships of their fine tuning between cardiovascular risk factors and the manifestation of diseases. Recently calculation of the number of research and development in these areas.

In the Figure 1 is shown dataset attributes all columns and rows.

```
dataset = pd.read_csv('cardio_train.csv', ';')
```

```
dataset.sample(10)
```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
47702	68105	15955	2	173	85.0	130	80	1	1	0	1	1	0
42749	61085	21912	2	167	80.0	140	90	1	1	0	0	0	1
8773	12518	23248	1	160	64.0	120	80	1	1	0	0	1	0
6340	9025	18299	1	156	104.0	120	80	1	1	0	0	1	1
56707	80954	15300	2	160	62.0	140	90	1	1	0	0	1	1
69834	99739	19776	1	156	59.0	120	80	2	1	0	0	0	0
16906	24167	17272	2	170	31.0	150	90	2	2	0	0	1	1
18950	27061	21158	1	168	64.0	120	80	1	1	0	0	0	0
36449	52065	20302	1	168	95.0	180	80	1	1	0	0	1	1
28611	40910	23455	2	176	85.0	140	90	2	2	0	0	1	1

Figure 1. Dataset description in Python

After dataset has been read in Python there is a method dataset.nunique() to check how many unique values are there in the each row. dataset.isna().sum() following method is used to verify are there any nill or empty rows. If yes, then we need to fill all null values by average sum of row and normalize the dataset. Then there is prepared dataset for further procedures.

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

Following code above is used to divide dataset into train and test. Furthermore, for X has chosen all columns except the last one, for y vice versa. It is important to separate them because our target to find people from dataset who has cardiovascular disease and doesn't have. There is a reason why we have 2 option only 1 and 0. 1 – yes, 0 –no.

There is used following piece of code to import train_test_split for separating dataset into test and train:

from sklearn.model_selection import train_test_split

From Figure 2 you are able to visualize it more clearly.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state = 1,stratify =y)
```

```
from sklearn.neighbors import KNeighborsClassifier
```

Figure 2. Importing train test split

After all these steps our dataset is ready to be modified and proceed.

2.2. Using machine learning algorithms for processing medical data

Machine learning is a process used by companies to turn raw data into useful information. By using software to look for patterns in large batches of data, businesses can learn more about their customers to develop more effective marketing strategies, increase sales and decrease costs. Machine learning depends on effective data collection, warehousing, and computer processing [1].

Machine learning involves exploring and analyzing large blocks of information to glean meaningful patterns and trends. It can be used in a variety of ways, such as database marketing, credit risk management, fraud detection, spam Email filtering, or even to discern the sentiment or opinion of users [2].

The most popular algorithms of machine learning are represented in Figure 3.

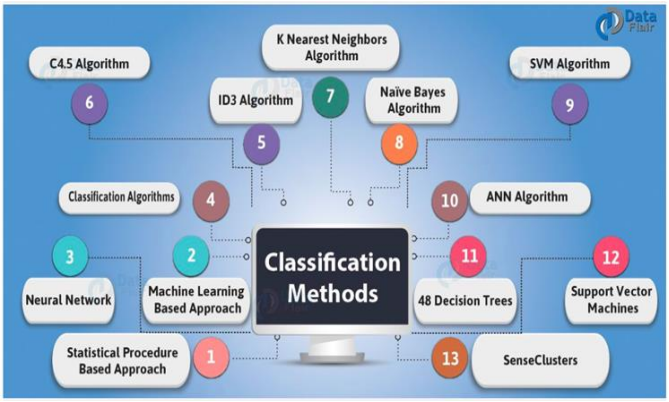

Figure 3. Classification Methods

Machine learning algorithms are able to be whether supervised or unsupervised.

Supervised learning: Algorithms that need a 'training' set of data to learn.

Unsupervised learning: Algorithms that don't need any training data to work properly.

Here are the main types of algorithm that is going to be used.

Classification: These algorithms put the existing data (or past data) into various 'classes' (hence classification) based on their attributes (properties) and use that classified data to make predictions.

Accuracy = TP+TN/TP+FP+FN+TN

Precision = TP/TP+FP

Recall = TP/TP+FN

F1 Score = 2*(Recall * Precision) / (Recall + Precision).

K Nearest Neighbor Algorithm. K-Nearest Neighbors, or KNN for short, is one of the simplest machine learning algorithms and is used in a wide array of institutions. KNN is a non-parametric, lazy learning algorithm. When we say a technique is non-parametric, it means that it does not make any assumptions about the underlying data. In other words, it makes its selection based off of the proximity to other data points regardless of what feature the numerical values represent. Being a lazy learning algorithm implies that there is little to no training phase. Therefore, we can immediately classify new data points as they present themselves [3].

Implementation in Python KNN algorithm. Let's explore our cardiovascular dataset in KNN algorithm in Figure 4.

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state = 1,stratify =y)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(51)
```

re is no big difference between 11th and 33th k neighbours only 57% is predicted

```
X_test[1:2]

array([[5.3647e+04, 1.9730e+04, 1.0000e+00, 1.4700e+02, 5.8000e+01,
        1.6500e+02, 1.0000e+02, 1.0000e+00, 1.0000e+00, 0.0000e+00,
        0.0000e+00, 1.0000e+00]])

knn.fit(X_train, y_train)

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=51, p=2,
                     weights='uniform')

y_preds = knn.predict(X_test)

for i,j in zip(y_test, y_preds):
    print('real: ', i, ' predicted: ',j)
```

Figure 4. KNN

The first step to import KNeighborsClassifier in order to analyze dataset and find optimal accuracy for dataset. At the beginning it was tried to use 11 and 33 neighbors. However, an accuracy was not as it was expected. As a result, it has been noticed that there is no big difference between 11th and 33th k neighbors only 57% is predicted.

	precision	recall	f1-score	support
Not cardio	0.59	0.59	0.59	7004
cardio	0.59	0.58	0.59	6996
accuracy			0.59	14000
macro avg	0.59	0.59	0.59	14000
weighted avg	0.59	0.59	0.59	14000

Figure 5. KNN, Confusion matrix and classification report

Figure 5 is shown accuracy score better predicted in the nearest neighbors 51 but in 55 it is getting worse. In order to predict better result, we need find the best option here. Before 51 it was tried all classifiers but accuracy was only between 56-57, the best result was shown only here, anyway 58 it is not tending to be a good prediction. As a result, you are able to see here accuracy result is about 59% which is not bad, but still need some good options to increase result.
Result: Accuracy of KNN is 59%.

Logistic Regression. Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign. Logistic regression transforms its output using the logistic sigmoid function to return a probability value [4].

Logistic regression defined as the «Sigmoid function» or also known as the «logistic function» instead of a linear function. Figure 6 is represented Sigmoid function, which defines Logistic regression.

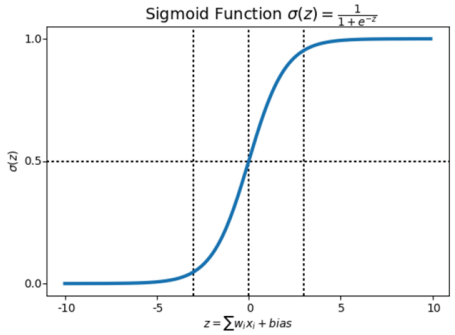

Figure 6. Sigmoid function

Implementation of Logistic Regression algorithm in Python. To explore our dataset by logistic Regression algorithm, there is been added method import LogisticRegression. Figure 7 is shown source code of importing methods and clusters in Python. Accuracy of logistic regression classifier on test set: 0.71(71%). This accuracy is better than KNN which has shown only 57%.

In the next Figure 8 we able to see confusion matrix and classifation reports.

The result of confusion matrix comparing KNN and Logistic Regression it is definitely better confusion matrix in Logistic regression 5255+1749>2323+1749 in KNN we are able to see following result in confusion matrix 4154+4082>2914+2850, as a result confusion matrix of Logistic Regression is found as the best. In Classification report

of Logistic Regression there is precision 69%, recall 75%, f1 score 72% higher than in KNN here following classifiers precision 59%, recall 59%, f1 score 59% constantly. Precision better predict for 10% defense, recall 16%, f1 score 13%.

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

logreg = LogisticRegression()
logreg.fit(X_train, y_train)

C:\Users\user\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be change
d to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)

y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))

Accuracy of logistic regression classifier on test set: 0.71
```

Figure 7. Logistic regression

	precision	recall	f1-score	support
0	0.68	0.73	0.70	7004
1	0.71	0.66	0.68	6996
accuracy			0.69	14000
macro avg	0.69	0.69	0.69	14000
weighted avg	0.69	0.69	0.69	14000

Figure 8. Logistic Regression, Confusion matrix and classification report

Let's visualize True and Positive Rate False Negative Rate as well in Figure 9.

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

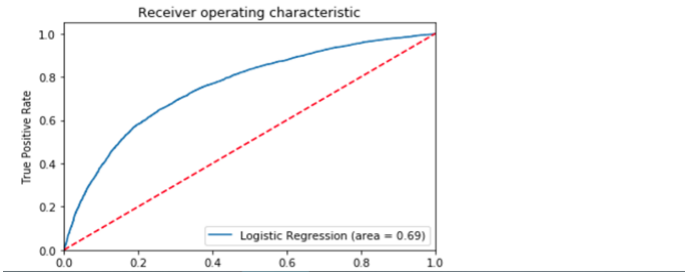

Figure 9. True Positive Rate and False Negative Rate of Logistic Regression

To sum up, I would probably choose in my case logistic regression, as a result it seems to make better prediction from 59% to 69%.

Result: Accuracy of Logistic Regression is 69%.

Support Vector Machine. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence [5].

In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values ([-1,1]) which acts as margin. In Formula 1 there is Hinge loss function.

$$c(x, y, f(x)) = \begin{cases} 0, & f\,y * f(x) \geq 1 \\ 1 - y * f(x), & else \end{cases} \quad (1)$$

Implementation in Python SVM algorithm. The earliest step we need to start with, is to import SVM. This method is used to call classifier SVC svclassifier = SVC ().

plt.scatter(X_train[:, 0], X_train[:, 4], c=y_train, cmap = 'spring'). Following code represents scatter plot that will show visualization of data, which is shown in Figure 10.

```
In [98]: plt.scatter(X_train[:, 0], X_train[:, 4], c=y_train, cmap = 'spring')
Out[98]: <matplotlib.collections.PathCollection at 0x20594c93848>
```

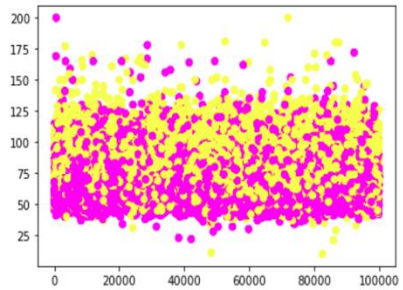

Figure 10. Scatter plot of dataset

For SVM is used 2 types of kernel sigmoid and rbf. Furthermore, it has got 2 accuracy results, confusion matrix and classifier report as well. Figure 11 is shown report of SVM rbf classifier report.

	precision	recall	f1-score	support
0	0.61	0.54	0.57	7004
1	0.59	0.66	0.62	6996
accuracy			0.60	14000
macro avg	0.60	0.60	0.60	14000
weighted avg	0.60	0.60	0.60	14000

Figure 11. SVM, rbf, Confusion matrix and classification report

As a result, it is clear that accuracy of rbf kernel for SVM is about 60% macro and weighted average the same 60% which better than in KNN which was 59% but worse than Logistic Regression which was 71%. Precision is 61% recall seems worse only 54% and support there are 7004 for 0 target. For target 1 result in precision worse 59% but recall 66% and f1-score 62%, support only 6996.

Now Figure 12 is presented Confusion matrix and classification report of SVM for sigmoid classifier. svclassifier1 = SVC(kernel='sigmoid') this following code is used to identify result for sigmoid.

	precision	recall	f1-score	support
0	0.51	0.50	0.51	7004
1	0.51	0.51	0.51	6996
accuracy			0.51	14000
macro avg	0.51	0.51	0.51	14000
weighted avg	0.51	0.51	0.51	14000

Figure 12. SVM, sigmoid, Confusion matrix and classification report

For cardiovascular dataset rbf classifier is better than sigmoid accuracy in rbf 0.508 accuracy in sigmoid 0.5995 comparing confusion matrix (sigmoid) 3535 3469] [3419 3577] correct predictions 7112>6888 negative predictions comparing confusion matrix (rbf) [3782 3222] [2385 4611] correct predictions 8393>5607 negative predictions precision 51% in sigmoid, precision 61% in rbf recall no big difference 50% and 54% in rbf f1 51% f1 57% in svm kernel rbf.

Result: Accuracy of SVM kernel = rbf is 60%.

Naive Bayes. A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem. In the following formula 2 you are able to see Bayes theorem about Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naïve [6].

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \quad (2)$$

Implementation in Python Naïve Bayes algorithm. In this Naïve Bayes algorithms are used three types of classifiers, such as Bernoulli Naive Bayes with following source code:

1. from sklearn.naive_bayes import BernoulliNB
2. from sklearn.model_selection import train_test_split
3. bnb = BernoulliNB(binarize=0.0)
4. After that has been used Multinomial Naïve Bayes with following source code:
5. from sklearn.naive_bayes import MultinomialNB
6. from sklearn.model_selection import train_test_split
7. mnb = MultinomialNB(alpha=0.01)
8. The last one Gaussian Naïve Bayes with following source code:
9. from sklearn.naive_bayes import GaussianNB
10. gnb = GaussianNB().

In Figure 12 is demonstrated accuracy 57% for Gaussian Naïve Bayes. Moreover, there is confusion matrix with not bad result and recall with 95% in classification result which the best comparing with others.

Accuracy:	precision	recall	f1-score	support
0	0.54	0.95	0.69	7004
1	0.78	0.19	0.30	6996
accuracy			0.57	14000
macro avg	0.66	0.57	0.49	14000
weighted avg	0.66	0.57	0.49	14000

Figure 13. Gaussian Naïve Bayes, Confusion matrix and classification report

In Figure 14 is demonstrated Bernoulli Naïve Bayes accuracy 52% with the same precision and f1-score as well. It seems Gaussian is better for the following dataset than Bernoulli.

Accuracy:	precision	recall	f1-score	support
0	0.51	0.82	0.63	7004
1	0.55	0.22	0.31	6996
accuracy			0.52	14000
macro avg	0.53	0.52	0.47	14000
weighted avg	0.53	0.52	0.47	14000

Figure 14. Bernoulli Naïve Bayes, Confusion matrix and classification report

Decision Tree. Decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in machine learning for deriving a strategy to reach a particular goal, it's also widely used in machine learning, which will be the main focus of this article. Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, you will need to trim it down for it to look beautiful [7].

Implementation of Decision Tree algorithm in Python. To implement DT in Python we need to import it via this code:

from sklearn.tree import DecisionTreeClassifier

In Figure 15 Accuracy result is shown as 64% which better than KNN and SVM, Naïve Bayes as well.

```
from sklearn.tree import export_graphviz

clf = DecisionTreeClassifier()

clf = clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)

y_pred

array([1, 0, 0, ..., 0, 0, 1], dtype=int64)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.641
```

Figure 15. DT accuracy

If we use DT entropy it is showed better result in Accuracy with 73% with following code:

clf = DecisionTreeClassifier (criterion="entropy", max_depth=3) in Figure 16.

```
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

clf = clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7288571428571429
```

Figure 16. DT(entropy) accuracy

In Figure 17 below it is represented Decision Tree of our dataset.

Result: Accuracy of Decision Tree is 73%.

Random Forest. Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction [7].

Implementation in Python Random Forest algorithm. As we know it starts from importing classifier with following code:

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=20, random_state=0)

regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)

In the Random Forest Mean Absolute Error: 0.3650785714285715

Mean Squared Error: 0.19626714285714283

Root Mean Squared Error: 0.4430204767921488.

```
graph.write_png('cardio.png')
Image(graph.create_png())
```

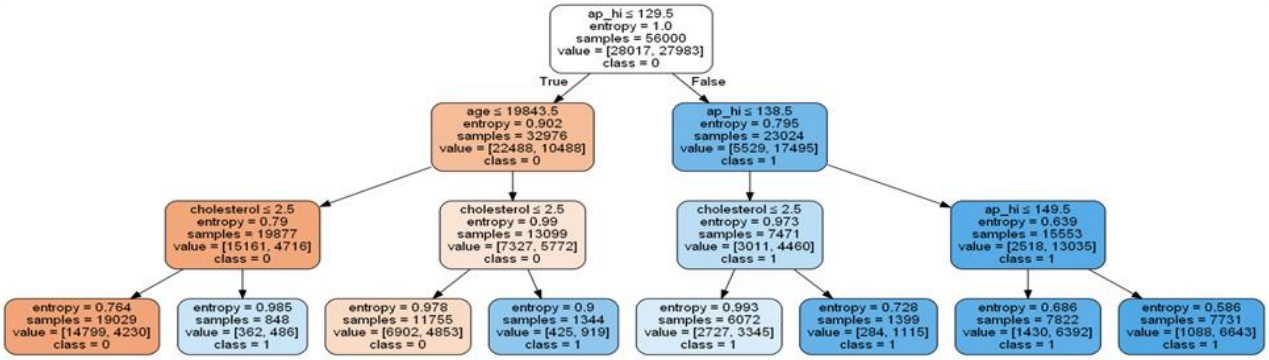

Figure 17. Visualization of DT

Random Forest gives the same accuracy as Decision Tree 73% in Figure 18 it is shown.

```
from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.728
```

Figure 18. RF accuracy

Result: Accuracy of Random Forest is 73%.

3. Results and discussion

In the previous section, we implemented the main algorithms of machine learning and tested them with a medical dataset. As a result of testing, the accuracy indicators of the algorithms turned out to be different. Their accuracy indicators are shown in Table 1.

Table 1. Accuracy of ML algorithms

№	Machine learning algorithm	Accuracy
1	KNeighborsClassifier	59%
2	Logistic Regression	69%
3	Support Vector Machine kernel=sigmoid	51%
4	Support Vector Machine kernel = rbf	60%
5	Gaussian Naïve Bayes	57%
6	Bernoulli Naïve Bayes	52%
7	Decision Tree	73%
8	Random Forest	73%

As shown in Table 1, the best accuracy algorithms are Decision Tree and Random Forest, accuracy=73%. During the test, two types of the Naïve Bayes method were considered and two different parameters of the Support Vector Machine algorithm were tested. But their accuracy is not higher than Decision Tree algorithm. Random Forest ia an ensemble implementation of Decision Tree algorithm.

4. Conclusions

This paper consists of all main methods and algorithms of Machine learning in order to correctly use a medical dataset and get some essential information from useless information. In this article were given following algorithms of Machine learning such as KNN, SVM, DT, RF, Naïve Bayes and Logistic Regression. The results were quite surprising. Accuracy of all algorithms were not less than 50% and not higher than 73%. The best result was demonstrated by Decision Tree and Random Forest they have shown the same result 73% and Logistic Regression had slightly fewer percentages as about 71%. The worst result has represented by Naïve Bayes only 57% whereas KNN had shown 59%.

To sum up, I would probably say that Decision Tree algorithm and Random Forest made perfect job for dataset cardiovascular_train.csv. I suppose, it happened because of datatypes and our target which was Boolean 1 and 0.

References

[1] Embi, P.J. & Payne, P.R. (2009). Clinical research informatics: challenges, opportunities and definition for an emerging do-

main. *Journal of the American Medical Informatics Association, 16(3), 316–327.* https://doi.org/10.1197/jamia.M3005
[2] Prokosch, H.U. & Ganslandt, T. (2009). Perspectives for medical informatics. Reusing the electronic medical record for clinical research. *Methods of information in medicine, 48(1), 38–44*
[3] Wasserman, R.C. (2011). Electronic medical records (EMRs), epidemiology, and epistemology: reflections on EMRs and future pediatric clinical research. *Academic pediatrics, 11(4), 280–287.* https://doi.org/10.1016/j.acap.2011.02.007
[4] Dean, B.B., Lam, J., Natoli, J.L., Butler, Q., Aguilar, D. & Nordyke, R.J. (2009). Review: use of electronic medical records for health outcomes research: a literature review. *Medical care research and review, 66(6), 611–638.* https://doi.org/10.1177/1077558709332440
[5] Tannen, R.L., Weiner, M.G., & Marcus, S.M. (2006). Simulation of the Syst-Eur randomized control trial using a primary care electronic medical record was feasible. *Journal of clinical epidemiology, 59(3), 254–264.* https://doi.org/10.1016/j.jclinepi.2005.08.008
[6] Williams, J.G., Cheung, W.Y., Cohen, D.R., Hutchings, H.A., Longo, M.F. & Russell, I.T. (2003). Can randomised trials rely on existing electronic data? A feasibility study to explore the value of routine data in health technology assessment. *Health technology assessment, 7(26), iii–117.* https://doi.org/10.3310/hta7260
[7] Yamamoto, K., Matsumoto, S., Tada, H., Yanagihara, K., Teramukai, S., Takemura, T. & Fukushima, M. (2008). A data capture system for outcomes studies that integrates with electronic health records: development and potential uses. *Journal of medical systems, 32(5), 423–427.* https://doi.org/10.1007/s10916-008-9147-7

Медициналық деректерді өңдеу үшін машиналық оқыту алгоритмдерін пайдалану

Г. Мукажанова[1], Ж. Алибиева[2*], А. Касенхан[2], Н. Мукажанов[2]

[1]*Инновациялық Еуразия Университеті, Павлодар, Қазақстан*
[2]*Satbayev University, Алматы, Қазақстан*

**Корреспонденция үшін автор: zh.alibiyeva@satbayev.university*

Аңдатпа. Бұл мақалада machine learning алгоритмдерінің салыстырмалы талдауы қарастырылады. Алгоритмдерді салыстру келесі деректер жиыны бойынша жүргізіледі: cardio_train.csv from kaggle.com (сілтеме: https://www.kaggle.com/sulianova/cardioascular-disease-dataset). Сонымен қатар, деректерді іздеу алгоритмдері cardio_train.csv дәлдігі бойынша ең жақсы алгоритмдерді анықтайды. Python 3.0 бағдарламалау тілінде орындалған процедураларды тексеру, матрица және жіктеу есебі, дәлдік көрсеткішін, еске түсіруді, f1 ұпайын және қолдауды көруге мүмкіндік береді. Сонымен қатар, осы мақалада келесі классификациялық модельдерді көруге болады: KNN алгоритмі, логистикалық регрессия, шешім ағашы, Random Forest, Naive Bayes және SVM. Нәтижесінде медициналық деректерді өңдеудің ең жоғары дәлдігі анықталады.

Негізгі сөздер: *медициналық мәліметтер базасы, деректер жинау, алгоритмдер, KNN алгоритмі, логистикалық регрессия, шешім ағашы, Random Forest, SVM, Naive Bayes*.

Использование алгоритмов машинного обучения для обработки медицинских данных

Г. Мукажанова[1], Ж. Алибиева[2*], А. Касенхан[2], Н. Мукажанов[2]

[1]*Инновационный Евразийский Университет, Павлодар, Казахстан*
[2]*Satbayev University, Алматы, Казахстан*

**Автор для корреспонденции: zh.alibiyeva@satbayev.university*

Аннотация. В статье рассматривается сравнительный анализ алгоритмов Machine learning для набора данных: cardio_train.csv от kaggle.com. (ссылка: https://www.kaggle.com/sulianova/cardiovascular-disease-dataset). Более того, с помощью алгоритмов machine learning будут обнаружены алгоритмы наилучшей точности для cardio_train.csv. Рассматривается процедура, разработанная на языке программирования Python 3.0, которая представляет собой confusion matrix и отчет о классификации, позволяет увидеть оценку точности, отзыв, оценку f1 и поддержку. Кроме того, в этой статье вы можете увидеть следующие модели классификации: алгоритм KNN, логистическая регрессия, дерево решений, Random Forest, наивный байесовский метод и SVM. В результате будет определена высочайшая точность обработки медицинских данных.

Ключевые слова: *набор медицинских данных, сбор данных, алгоритмы, KNN алгоритм, логистическая регрессия, древо решений, Random Forest, SVM, наив байесовский метод*.

Received: 13 December 2022
Accepted: 16 March 2023
Available online: 31 March 2023